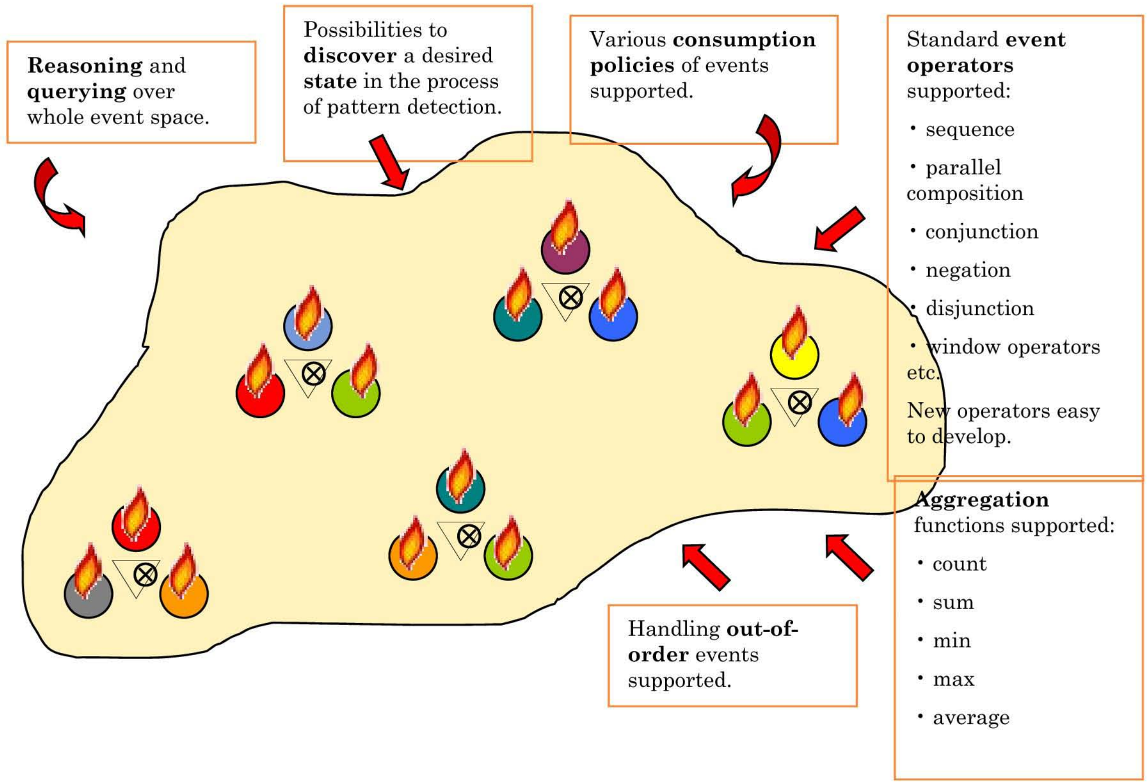
ETALIS Language for Complex Event Processing

- **Data-driven with declarative semantics complex event processing**
- Complex events are *derived* from simpler events by means of *deductive* rules
- **Combines detection of complex events and reasoning over states**
- ETALIS is implemented in Prolog and runs on XSB, YAP, SWI, and SICStus.



ETALIS LANGUAGE FOR COMPLEX EVENTS

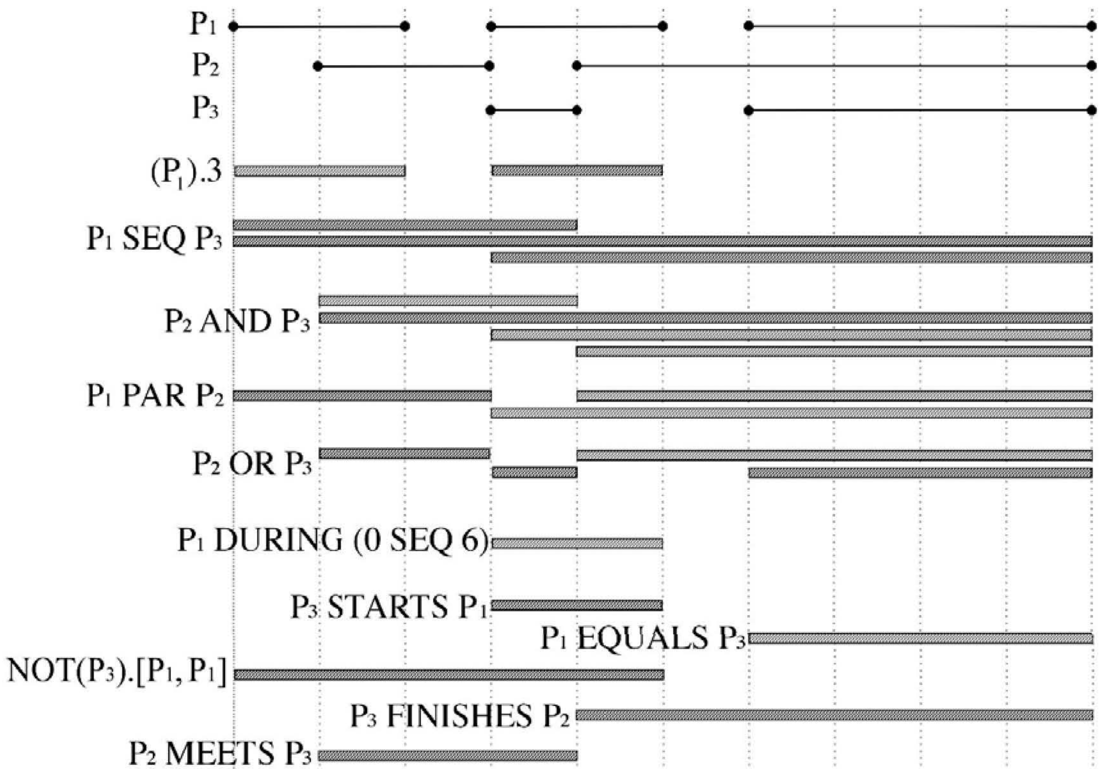
event patterns

$$P ::= \mathtt{pr}(t_1,\ldots,t_n) \mid P \text{ WHERE } t \mid q \mid (P).q \mid P \text{ BIN } P \mid \text{NOT}(P).[P,P]$$

event rule

$$\mathtt{pr}(t_1,\ldots,t_n) \leftarrow p$$

Composition Operators



event stream $\varepsilon : Ground \to 2^{\mathbb{Q}^+}$

variable assignment is a mapping $\mu : Var \to Con$

interpretation $\mathscr{I} : Ground \to 2^{\mathbb{Q}^+ \times \mathbb{Q}^+}$

$$q_1 \leq q_2 \text{ for every } \langle q_1, q_2 \rangle \in \mathscr{I}(g) \text{ for all } g \in Ground$$

Given an event stream ε, an interpretation $\mathscr{I}$ is called a *model* for a rule set $\mathscr{R}$ – written as $\mathscr{I} \models_\varepsilon \mathscr{R}$ – if the following conditions are satisfied:

C1 $\langle q, q \rangle \in \mathscr{I}(g)$ for every $q \in \mathbb{Q}^+$ and $g \in Ground$ with $q \in \varepsilon(g)$

C2 for every rule $atom \leftarrow pattern$ and every variable assignment μ we have

$$\mathscr{I}_\mu(atom) \subseteq \mathscr{I}_\mu(pattern) \text{ where } \mathscr{I}_\mu \text{ is inductively defined as}$$

pattern	$\mathscr{I}_\mu$(pattern)
$\mathtt{pr}(t_1, \ldots, t_n)$	$\mathscr{I}(\mathtt{pr}(\mu^*(t_1), \ldots, \mu^*(t_n)))$
p WHERE t	$\mathscr{I}_\mu(p)$ if $\mu^*(t) = true$ $\emptyset$ otherwise.
q	$\{\langle q, q \rangle\}$ for all $q \in \mathbb{Q}^+$
$(p).q$	$\mathscr{I}_\mu(p) \cap \{\langle q_1, q_2 \rangle \mid q_2 - q_1 = q\}$
p_1 SEQ p_2	$\{\langle q_1, q_4 \rangle \mid \langle q_1, q_2 \rangle \in \mathscr{I}_\mu(p_1)$ and $\langle q_3, q_4 \rangle \in \mathscr{I}_\mu(p_2)$ and $q_2 < q_3\}$
p_1 AND p_2	$\{\langle \min(q_1, q_3), \max(q_2, q_4) \rangle \mid \langle q_1, q_2 \rangle \in \mathscr{I}_\mu(p_1)$ and $\langle q_3, q_4 \rangle \in \mathscr{I}_\mu(p_2)\}$
p_1 PAR p_2	$\{\langle \min(q_1, q_3), \max(q_2, q_4) \rangle \mid \langle q_1, q_2 \rangle \in \mathscr{I}_\mu(p_1)$ and $\langle q_3, q_4 \rangle \in \mathscr{I}_\mu(p_2)$ and $\max(q_1, q_3) < \min(q_2, q_4)\}$
p_1 OR p_2	$\mathscr{I}_\mu(p_1) \cup \mathscr{I}_\mu(p_2)$
p_1 EQUALS p_2	$\mathscr{I}_\mu(p_1) \cap \mathscr{I}_\mu(p_2)$
p_1 MEETS p_2	$\{\langle q_1, q_3 \rangle \mid \langle q_1, q_2 \rangle \in \mathscr{I}_\mu(p_1)$ and $\langle q_2, q_3 \rangle \in \mathscr{I}_\mu(p_2)\}$
p_1 DURING p_2	$\{\langle q_3, q_4 \rangle \mid \langle q_1, q_2 \rangle \in \mathscr{I}_\mu(p_1)$ and $\langle q_3, q_4 \rangle \in \mathscr{I}_\mu(p_2)$ and $q_3 < q_1 < q_2 < q_4\}$
p_1 STARTS p_2	$\{\langle q_1, q_3 \rangle \mid \langle q_1, q_2 \rangle \in \mathscr{I}_\mu(p_1)$ and $\langle q_1, q_3 \rangle \in \mathscr{I}_\mu(p_2)$ and $q_2 < q_3\}$
p_1 FINISHES p_2	$\{\langle q_1, q_3 \rangle \mid \langle q_2, q_3 \rangle \in \mathscr{I}_\mu(p_1)$ and $\langle q_1, q_3 \rangle \in \mathscr{I}_\mu(p_2)$ and $q_1 < q_2\}$
NOT$(p_1).[p_2, p_3]$	$\mathscr{I}_\mu(p_2$ SEQ $p_3) \setminus \mathscr{I}_\mu(p_2$ SEQ p_1 SEQ $p_3)$

Given an interpretation $\mathscr{I}$ and some $q \in \mathbb{Q}^+$, we let $\mathscr{I}|_q$ denote the interpretation defined by $\mathscr{I}|_q(g) = \mathscr{I}(g) \cap \{\langle q1, q2 \rangle \mid q2 - q1 \leq q\}$.

Given two interpretations $\mathscr{I}$ and $\mathscr{J}$, we say that $\mathscr{I}$ is *preferred* to $\mathscr{J}$ if there exists a $q \in \mathbb{Q}^+$ with $\mathscr{I}|_q \subset \mathscr{J}|_q$.

A model $\mathscr{I}$ is called *minimal* if there is no other model preferred to $\mathscr{I}$.

Incremental Detection of Composed Events – Compiled rules in Prolog

Algorithm 5.2 Conjunction.

Input: event binary goal $ie_1 \leftarrow a$ AND b.

Output: event-driven backward chaining rules for AND operator.

Each event binary goal $ie_1 \leftarrow a$ AND b. is converted into: {

$a(T_1,T_2) : -for\_each(a,1,[T_1,T_2]).$

$a(1,T_3,T_4) \ : \ -goal(a(\_,\_),b(T_1,T_2),ie_1(\_,\_)), retract(goal(a(\_,\_),\ b(T_1,T_2),ie_1(\_,\_))),$

$T_5 \ = \ min\{T_1,T_3\}, T_6 \ = \ max\{T_2,T_4\},$

$ie_1(T_5,T_6).$

$a(2,T_1,T_2) : -\neg(goal(a(\_,\_),b(T_1,T_2),ie_1(\_,\_))),$

$assert(goal(b(\_,\_),a(T_1,T_2),ie_1^{(}\_,\_))).$

$b(T_3,T_4) : -for\_each(b,1,[T_3,T_4]).$

$b(1,T_3,T_4) : -goal(b(\_,\_),a(T_1,T_2),ie_1(\_,\_)), retract(goal(b(\_,\_),$

$a(T_1,T_2),ie_1(\_,\_))), T_5 = min\{T_1,T_3\}, T_6 = max\{T_2,T_4\},$

$ie_1(T_5,T_6).$

$b(2,T_1,T_2) : -\neg(goal(b(\_,\_),a(T_1,T_2),ie_1(\_,\_))), assert(goal(a(\_,\_),$

$b(T_1,T_2),ie_1(\_,\_))).$

}

“The Fast Flower Delivery Use Case”, *accompanying the book “Event Processing In Action”, by Opher Etzion and Peter Niblett, Manning Publications, 2009.*

http://code.google.com/p/etalis/wiki/Fast_Flower_Delivery_Use_Case

All 5 phases: bid, assignment, delivery process, ranking evaluation, activity monitoring.

```
% Phase 1: Bid Phase
% Multiplier: multiply the event "delivery_request_enriched" for each driver
% delivery_request_enriched_multiplied/6
delivery_request_enriched_multiplied(DeliveryRequestId,DriverId,StoreId,
        ToCoordinates,DeliveryTime,MinRank)<-
    delivery_request_enriched(DeliveryRequestId,StoreId,ToCoordinates,
        DeliveryTime,MinRank) event_multiply
    driver_record(DriverId,_Ranking).

% gps_location_translated/3
gps_location_translated(DriverId,Rank,Region)<-
    gps_location(DriverId,coordinates(SNHemisphere,Latitude,
        EWHemisphere,Longitude)) where
        ( driver_record(DriverId,Rank),
        gps_to_region(coordinates(SNHemisphere,Latitude,
            EWHemisphere,Longitude),Region) ).

% bid_request/5
bid_request(DeliveryRequestId,DriverId,StoreId,ToCoordinates,DeliveryTime)<
-
    ( delivery_request_enriched_multiplied(DeliveryRequestId,DriverId,
            StoreId,ToCoordinates,DeliveryTime,MinRank) and
        gps_location_translated(DriverId,Rank,Region) )
    where ('=<'(MinRank,Rank), gps_to_region(ToCoordinates,Region)).
…
```

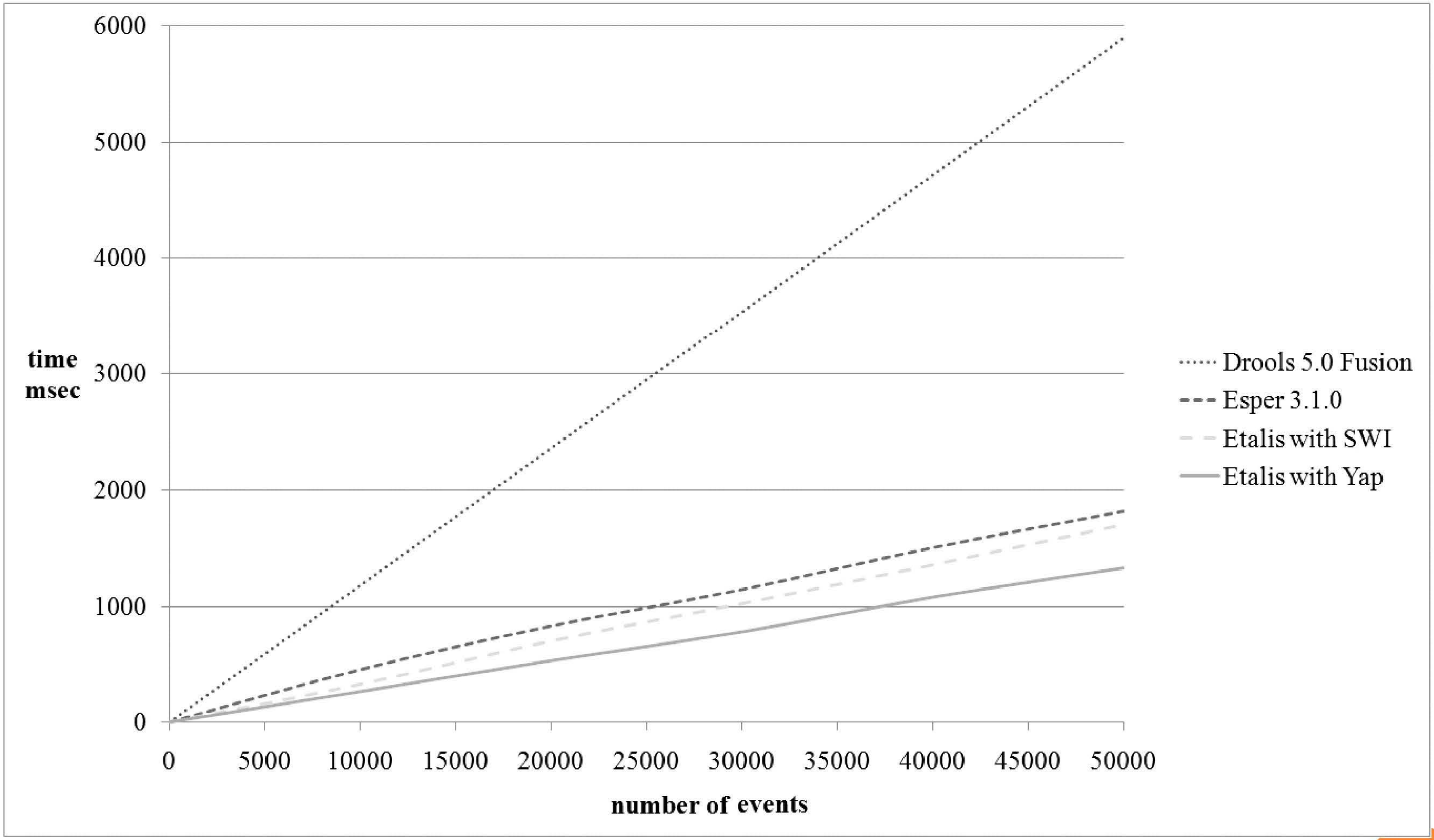
```
% Phase 2: Assignment Phase
% startAssignment/4
exceptionAlarm(startAssignment(DeliveryRequestId,StoreId,ToCoordinates,
        DeliveryTime),Time)<-
    delivery_request_enriched(DeliveryRequestId,StoreId,ToCoordinates,
        DeliveryTime,_MinRank) where (start_assignment_time(Time)).

% start_automaticAssignment/4
start_automaticAssignment(DeliveryRequestId,StoreId,ToCoordinates,
        DeliveryTime)<-
    startAssignment(DeliveryRequestId,StoreId,ToCoordinates,DeliveryTime)
    where store_record(StoreId,_MinRank,automatic).

% start_manualAssignment/4
start_manualAssignment(DeliveryRequestId,StoreId,ToCoordinates,DeliveryTime)<-
    startAssignment(DeliveryRequestId,StoreId,ToCoordinates,DeliveryTime)
    where store_record(StoreId,_MinRank,manual).

consumable_pick_first(DeliveryRequestId,StoreId,ToCoordinates,DeliveryTime,
        MinRank)<-
    delivery_request_enriched(DeliveryRequestId,StoreId,ToCoordinates,
    DeliveryTime,MinRank) where store_record(StoreId,_MinRank,automatic).

assignment(DeliveryRequestId,StoreId,ToCoordinates,DeliveryTime,DriverId,
        ScheduledPickupTime)<-
    ((consumable_pick_first(DeliveryRequestId,StoreId,ToCoordinates,
        DeliveryTime,MinRank) seq
    delivery_bid(DeliveryRequestId,DriverId,CurrentCoordinates,
        ScheduledPickupTime) ) and
    start_automaticAssignment(DeliveryRequestId,StoreId,ToCoordinates,
        DeliveryTime)).
…
```



time
msec
6000
5000
4000
3000
2000
1000
0
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
number of events
Drools 5.0 Fusion
Esper 3.1.0
Etalis with SWI
Etalis with Yap